Software Design Document

Los Angeles County Fleet Management System
(LACFMS)

Software Design Document
Version 1.0

Prepared by

Flinner, Patrick
Ghazarian, Avik
Karapetyan, Robert
Romero, Aaron
Shimauchi, Ken
Valadez, Benjamin

California State University of Los Angeles
December 1, 2017
Los Angeles County Fleet Management System

Software Design Document
Version 1.0

Prepared By:

Flinner, Patrick
Ghazarian, Avik
Karapetyan, Robert
Romero, Aaron
Shimauchi, Ken
Valadez, Benjamin

Approved By:

______________________________________  _____________
Dr. Russ Abbott                           Date

______________________________________  _____________
Dr. Sue Lim                               Date

______________________________________  _____________
Los Angeles County Liaison                Date

System Design Document for Los Angeles County Fleet Management System,
by Flinner, Ghazarian, Karapetyan, Shimauchi, Valadez, Fall 2017.
<table>
<thead>
<tr>
<th>Table of Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision History</td>
</tr>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>1.1. Purpose</td>
</tr>
<tr>
<td>1.2. Document Conventions</td>
</tr>
<tr>
<td>1.3. Intended Audience and Reading Suggestions</td>
</tr>
<tr>
<td>1.4. System Overview</td>
</tr>
<tr>
<td>1.5. References</td>
</tr>
<tr>
<td>2. Design Considerations</td>
</tr>
<tr>
<td>2.1. Assumptions and dependencies</td>
</tr>
<tr>
<td>2.2. General Constraints</td>
</tr>
<tr>
<td>2.3. Goals and Guidelines</td>
</tr>
<tr>
<td>2.4. Development Methods</td>
</tr>
<tr>
<td>3. Architectural Strategies</td>
</tr>
<tr>
<td>3.1. Architectural Design</td>
</tr>
<tr>
<td>3.2. User Interface</td>
</tr>
<tr>
<td>4. System Architecture</td>
</tr>
<tr>
<td>4.1. Architectural Overview</td>
</tr>
<tr>
<td>4.2. System Overview</td>
</tr>
<tr>
<td>5. Policies and Tactics</td>
</tr>
<tr>
<td>5.1. Specific Products Used</td>
</tr>
<tr>
<td>5.2. Requirements traceability</td>
</tr>
<tr>
<td>5.3. Testing the software</td>
</tr>
<tr>
<td>5.4. Hardware Performance Testing</td>
</tr>
<tr>
<td>5.5. Engineering trade-offs</td>
</tr>
<tr>
<td>5.6. Guidelines and conventions</td>
</tr>
<tr>
<td>5.7. Protocols</td>
</tr>
<tr>
<td>5.8. Maintaining the software</td>
</tr>
<tr>
<td>5.9. Interfaces</td>
</tr>
<tr>
<td>5.10. System's deliverables</td>
</tr>
<tr>
<td>5.11. Abstraction</td>
</tr>
<tr>
<td>6. Detailed System Design</td>
</tr>
<tr>
<td>6.x  Name of Module</td>
</tr>
<tr>
<td>6.x.1 Responsibilities</td>
</tr>
<tr>
<td>6.x.2 Constraints</td>
</tr>
<tr>
<td>6.x.3 Composition</td>
</tr>
<tr>
<td>6.x.4 Uses/Interactions</td>
</tr>
<tr>
<td>6.x.5 Resources</td>
</tr>
<tr>
<td>6.x.6 Interface/Exports</td>
</tr>
</tbody>
</table>
7. Detailed Lower level Component Design
   7.x Name of Class or File
   7.x.2 Processing Narrative(PSPEC)
   7.x.3 Interface Description
   7.x.4 Processing Detail
   7.x.4.1 Design Class Hierarchy
   7.x.4.2 Restrictions/Limitations
   7.x.4.3 Performance Issues
   7.x.4.4 Design Constraints
   7.x.4.5 Processing Detail For Each Operation

8. User Interface
   8.1. Overview of User Interface
   8.2. Screen Frameworks or Images
   8.3. User Interface Flow Model

9. Database Design
10. Requirements Validation and Verification
12. Hardware Schematics
13. References
### Revision History

<table>
<thead>
<tr>
<th>Name</th>
<th>Date</th>
<th>Reason For Changes</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>LACFMS</td>
<td>1/18/2018</td>
<td>Initial draft in progress</td>
<td>1.0</td>
</tr>
</tbody>
</table>

1. **Introduction**

Parks and Recreation has a fleet over 500 vehicles of different makes and models. Currently all vehicles use paper logs for the purpose of tracking vehicle usage and safety checks. This process can be time consuming for the driver, service personnel, and relies on the accuracy of manually supplied information.

The Los Angeles County Department of Parks and Recreation has teamed up with California State University: Los Angeles for a multi-year project to combat these issues.

An automated system will be developed to retrieve information from County motor vehicles. This system is intended to eliminate the paper based process, improve reporting, and enhance vehicle maintenance capabilities. An authentication component is also part of the interface for every vehicle. Only employees of Los County Parks and Recreation should be able to drive the vehicles, so a lockout mechanism will be applied based on the employee’s identification badge. Implementation of this system should reduce paper waste footprint, improve County resources, improve control over maintenance costs, and a simplified process for authorizing and reporting usage of County vehicles.

1.1 **Purpose**

This document contains the software design specification for LACFMS application project. The document is similar to IEEE specification 1016, with slight modifications to subsections which are for clarification purposes. All information here is prioritized and has not been committed for release. This is version 1.0 revision N/A at this time. Further, this document is part of another document, the Software Requirements Specification which will be the governing requirements for the implementation described here. Currently, this design specification will be broken up into three major phases and is currently focusing on the the edge and platform phase. The Edge phase of the project will include, hardware prototyping and data ingestion of sensory data, which will be stored locally on Edge devices. The Platform phase of the project will focus on the communication between Edge devices to infrastructure, secure messaging, and data accessibility and storage. Lastly, the Enterprise portion will focus on predictive analytics on stored datas.

1.2 **Document Conventions**

This document follows MLA Format. Bold-faced text has been used to emphasize section headings of each sub-section . Italicizing text words is to point out words that have been defined in the glossary. Every requirement statement in this document will have its own priority.

1.3 **Intended Audience and Reading Suggestions**

This document is intended for developers, testers, and project managers who wish to read it. This document is arranged according to standard IEEE specification 1016. **Bold** typeface is used to highlight subsections. **Italics** are used to emphasis words that have been defined in
Appendix A. Further, the software document is intended to be used by members of the project team that will implement and verify the correct functioning of the system.

1.4 System Overview

The Los Angeles County fleet management system abbreviated (LACFMS) is a software application used to monitor and report vehicle usage. The edge, is comprised of hardware components and is managed by a data flow file management technology. The LACFMS shall interface with these hardware components to extract, store and transmit vehicle diagnostics and GPS data using IoT practices. The web application shall act as an intermediate allowing employees of Los Angeles County to submit information as a replacement to the County Vehicle Mileage and Safety Check Form. Employees shall authenticate using a web interface. Only Managers will be permitted to view detailed trip descriptions and trip visualizations. This document is currently in the Edge phase of implementation which focuses on data ingestion and data storage, while the Platform and Enterprise phases will be postponed until the Edge is fully functional. The Platform interface, Apache Nifi and Minifi should be configured in this phase to communicate with one another using a web-socket scheme to transfer data securely from edge device to infrastructure. Data transferred should be stored in a reliable data storage for accessibility. The Enterprise phase should integrate data analytics.
2. Design Considerations

2.0.1 RFID device operating on 125kHz which enables and disables vehicle without interrupting the vehicle starter may not be feasible.

2.0.2 Storing sensory data on a raspberry pi 3 sim card, for an extended period of time may overload the capacity of the storage device.

2.0.3 Transmitting large files may cause the devices to overheat due to load.

2.0.4 Transmitting large files may take considerable amount of time while at County Facility.

2.0.5 Loss of power may corrupt file system.

Testing Issues of concern

Currently, testing a file flow manager technology at the edge.

2.1 Assumptions and Dependencies

Software Dependencies

2.1.1 Software libraries currently used by edge devices may need to be reconfigured on newer devices due to hardware changes and software changes.

- Raspberry PI3 Model B uses bluetooth on serial port ttyAMA0, which interferes with NEO 6M GPS hardware. Consequently, NEO 6M GPS hardware needs to be configured on a different serial port, for example, use serial port ttyS0, or another available serial configuration.

2.1.2 Operating System for Edge Devices Raspberry PI 3 Model B

https://www.raspberrypi.org/downloads/ version 2.4.4

2.1.3 Python-OBD library


2.1.4 Pynmea2 library

https://github.com/Knio/pynmea2
2.1.5 GPSD library
https://github.com/wdalmut/libgps

2.1.6 Apache Nifi version 1.40 (current)
https://cwiki.apache.org/confluence/display/NIFI/Release+Notes

2.1.7 Apache MiniFi version 0.20(current)
https://cwiki.apache.org/confluence/display/MINIFI/Release+Notes

2.1.8 Raspberry Pi3 Model B device for sensory data collection and storage uses a file flow management technology Apache MiniFi, this technology is better equipped on Edge devices than Apache Nifi due to the constrained hardware limitations. Apache MiniFi, is a reduced set of Apache Nifi, as the name suggests it requires less hardware resources and is better suited for hardware constrained devices.

2.1.9 Operating System on raspberry Pi 3 Model B is raspbian, this software is currently open-source but may change significantly enough over time to have an effect on used libraries and software dependencies. This may have a negative impact on Maintainability, of the products life cycle.

2.2 General Constraints

2.2.1. RFID capabilities which enable or disable vehicle starter are not feasible, all available resources suggest that such a feature would require interrupting the vehicle starter system which requires vehicles to have wires cut in order to successfully implement.

2.2.2. Apache Minifi is to be housed on Edge devices which it is designed to do; however, considerations need to be taken to ensure proper shutdown and to ensure cold start operations do not corrupt the files system of the device.

2.2.3. Software environment on the Edge needs to support batch transmission in the event that streaming data is not possible due to network connectivity interruptions.

- Storing sensory data locally for an unknown time for an available network to become available may not be feasible, considerations on stored data formats need to be tested to see what data format uses the least amount of storage capacity.

2.2.4. Considerations must be taken when transmitting large amounts of data.
2.2.4.1 Overheating on Edge Device, time frame of how long it will take to transmit.

2.2.4.2 How long will it take for the transfer of data from Edge devices to Infrastructure.

2.2.5 Difficulty may arise in targeting a software implementation for Toyota Prius 2007. Feedback is necessary from the vehicles that are currently being tested.

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

- Hardware or software environment
- End-user environment
- Availability or volatility of resources
- Standards compliance
- Interoperability requirements
- Interface/protocol requirements
- Data repository and distribution requirements
- Security requirements (or other such regulations)
- Memory and other capacity limitations
- Performance requirements
- Network communications
- Verification and validation requirements (testing)
- Other means of addressing quality goals
- Other requirements described in the requirements specification

You will not need to include all of these. Only the ones that will influence the design of your software

2.3 Goals and Guidelines Requirements

2.3.1 System software has an estimated delivery date TBD.

2.3.2 The Software requirements specification is a guideline to the implementation of the SRD.
2.4 Development Methods

Agile Development methods were used in order to promote rapid development in a constrained time frame. While, the waterfall paradigm approach use a similar iterative process as Agile methodologies. The tendency is that waterfall methodology tend to be more rigid and less flexible which causes software development and deployment to be slower and costly. This project employs a flexible methodology, where prototyping hardware has significant impact on design and deadline. Common practice is as follows.

*font change*

1. Configure
2. Build
3. Design
4. Release
5. Test
3. Architectural Strategies

The system architecture has been divided into three main build phases.

3.0.1 The Edge phase, includes hardware devices at the edge for data ingestion and temporary storage to support batch transmission.

3.0.2 The Platform phase, includes platform technology, Apache Nifi to be configured to communicate with IoT gateway, Raspberry Pi 3 Model B using Apache MiniFi. These technologies assist data flow, simplifying the transit of stored data from ingestion to Infrastructure.

3.0.3 The Enterprise phase
   Apache Nifi configured to transmit to Apache Spark or Maximo.(This is still being evaluated!)

3.0.4 Design Decision

  3.0.4.1 Design Decisions were made to use a file flow management technology, Apache Nifi in conjunction with Apache miniFi. These technologies will be used to manage the flow of data from Edge devices to Infrastructure, e.g. The Platform.

  3.0.4.2 GPS, is collected every second to reduce storage capacity.

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

  · Use of a particular type of product (programming language, database, library, etc. ...)
  · Reuse of existing software components to implement various parts/features of the system
  · Future plans for extending or enhancing the software
· User interface paradigms (or system input and output models)
· Hardware and/or software interface paradigms
· Error detection and recovery
· Memory management policies
· External databases and/or data storage management and persistence
· Distributed data or control over a network
· Generalized approaches to control
· Concurrency and synchronization
· Communication mechanisms
· Management of other resources

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

3.2 User Interfaces
4. System Architecture

4.1 Architectural Overview

**The Edge**

The Edge represents hardware components used for data ingestion. The Edge devices operates two fold, gathering sensory data; gps and vehicle diagnostics data. This data is then stored and made ready for batch processing or streaming. The microdevice is configured to operate as a server using (Apache MiniFi) which then broadcasts this data over a secure messaging protocol (mqtt encryption) and is consumed in another instance of the File flow manager technology (Apache Nifi). This is currently in an experimental phase of implementation.

**The Platform**

The platform consumes data being sent by devices on the edge. Data transmission should include a secure message protocol, mqtt is preferred. Data stored in platform should be accessible through a web interface.

**The Enterprise**

The enterprise phase should integrate data analytics integration on stored datas.
4.2 System Overview

The Fleet Management system described above has been Modularized for simplicity. More modules will be added or removed throughout the implementation phases. This is likely to evolve as the project evolves, and is no way representative of a finished product.
4.1 GPS Module provides GPS, formatted in NMEA 0183, for more information on NMEA. [https://en.wikipedia.org/wiki/NMEA_0183](https://en.wikipedia.org/wiki/NMEA_0183) Data will be stored locally on Edge devices, which will later be transmitted in a batch process.

4.2 OBDII Scanner requests vehicle diagnostics data from ECU. The ECU responds with Vehicle Subsystem data. OBD II scanner is capable of reading all vehicle engine protocols. Data is stored in JSON format locally which can be used by Apache miniFi to transmit.

4.3 Login Module provides a user interface where driver and manager can view web content. Allows authorization for Los Angeles Park and Recreation to block non-users from viewing specific web content.

4.3.1 TBD

4.3.2 TBD
4.4 Driver Module automates County Vehicle Mileage and Safety Check Form. Allows drivers to input this information in a digital manner. Data is stored in a persistent database which can be accessible.

4.5 Manager Module is a web page which only managers can view. This web page displays a list of trips with driver information. A map visualization of each trip which includes a route along with a speed or velocity for each route taken.

4.6 Database Module is used for persistent storage for Driver input information, along with GPS data. Data stored in this module is for the creation of a Map Visualization, and also made accessible for administrative purposes.

Other Modules will be added as need be. Currently, this represents the general structure.

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it). Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.

Level 1 Data Flow Diagrams (DFD) and Control Flow Diagrams (CFD) should probably go here.
Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

5. Policies and Tactics
Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless impact the details of the interface and/or implementation of various aspects of the system. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Such decisions might concern (but are not limited to) things like the following (Must include 5.1, 5.2, and 5.3. The rest of these categories or custom ones can be added as needed.):

5.1 Choice of which specific products used

Software

5.1.1 Visual Studio 2017, .NET Core Technologies Compiler
5.1.2 Angular Framework version 2.0
5.1.3 JRE 1.8 version
5.1.4 Java version 8
5.1.5 GPSD library for raspbian OS
5.1.6 Pynmea2 library for raspbian OS
5.1.7 Sqlite for raspbian OS
5.1.8 OBD library for python for raspbian OS
5.19 Apache Nifi
5.1.10 Apache MiniFi

Hardware

5.1.11 GPS NEO 6M Ublox
5.1.12 Raspberry Pi3 Model B
5.1.13 OBDII Scanner

5.2 Plans for ensuring requirements traceability

Traceability Diagram Here

<...>

5.3 Plans for testing the software
Scenario Testing for Each Use Case

5.3.1 To Be Determined TBD

5.4. Hardware Performance Testing

5.4.1 GPS Processing Test
   - Performance Testing on Edge Device Load
     5.4.1 gps processing at 1Hz or 1 cycle per second.
     5.4.2 gps processing at 5Hz or 5 cycles per second.
   - Trade-Offs

Advantage to accuracy—There is no beneficial gain when dealing with car vehicle velocity. Positional data is accurate and sufficient at 1 cycle per second, or 1Hz.
Testing was conducted with Ublox software which allows configuration changes at the hardware level. See, https://www.u-blox.com/en/product/u-center-windows as a reference.

- Advantage to performance
  None- 1Hz gps sensory collection is sufficient.

Trade-offs exist there is more benefits to reducing storage capacity by collecting gps at 1 second versus 5 collections per second. Theoretically, reducing storage capacity 5x over the same time frame.

Overview of hardware limitations

- Velocity accuracy  0.1m/s
- Heading accuracy  0.5 degrees
- Operational Limits Dynamics 4 g
- Altitude  50,000 m (meters)
- Velocity  500 m/s (meters per second)

5.4.2 OBD II Scanner Test

…Describe…

5.# Engineering trade-offs
  …Describe…

5.# Coding guidelines and conventions
  …Describe…

5.# The protocol of one or more subsystems, modules, or subroutines
  …Describe…

5.# The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality
  …Describe…

5.# Plans for maintaining the software
  …Describe…

5.# Interfaces for end-users, software, hardware, and communications
  …Describe…
5. Hierarchical organization of the source code into its physical components (files and directories).

…Describe…

5. How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

…Describe…

5. Describe tactics such as abstracting out a generic Database Interface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the Database Interface class.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (subsection as its corresponding software component in one of these other sections.
6. Detailed System Design

In progress!
Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:
This is where Level 2 (or lower) DFD’s will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

6.x Name of Component (Module)

6.x.1 Login Component

The component allows user to be permitted to view protected web content. This component provides several tasks. It allows user to input data into a form and submit data to a
database. This allows drivers to submit data for each work request. Similar to the County Vehicle Mileage and Safety Check Form.

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

6.2 Constraints

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, postconditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

6.3 Composition

A description of the use and meaning of the subcomponents that are a part of this component.

6.4 Uses/Interactions

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclasses, and metaclasses.

6.5 Resources

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.6 Interface/Exports

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).
Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.
7. Detailed Lower level Component Design
Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. “7.1.3 The ABC Package”). Note that there isn't necessarily a one-to-one correspondence between packages and components.

7.x Name of Class or File

7.x.1 Classification
The kind of component, such as a subsystem, class, package, function, file, etc.

7.x.2 Processing Narrative (PSPEC)
A process specification (PSPEC) can be used to specify the processing details

7.x.3 Interface Description

7.x.4 Processing Detail

7.x.4.1 Design Class Hierarchy
Class inheritance: parent or child classes.

7.x.4.2 Restrictions/Limitations

7.x.4.3 Performance Issues

7.x.4.4 Design Constraints

7.x.4.5 Processing Detail For Each Operation
8. Database Design

**Database**
- + name : String
- + version:

**Driver Table**
- +int: EmployeeID (Primary Key)
- +varchar: First Name
- +varchar: Last Name
- +varchar: Middle Name
- +date: Date of Birth
- +varchar: Phone Number
- +varchar: License Number

**Trip Table**
- +int: EventID (Primary Key)
- +bigint: initialOdometer
- +bigint: final Odometer
- +varchar: Project Title
- +varchar: Purpose
- +varchar: Function (see spreadsheet)
- +varchar: TaskOrder

**Trip Table (cont)**
- +datetime: startuptime
- +datetime: endingtime
- +longtext: tripPath
- +varchar: VehicleID (Foreign Key)
- +int: EmployeeID (Foreign Key)

**Vehicle Table**
- +varchar: Vehicle ID Number
- +varchar: Make
- +varchar: Model
- +varchar: License Plate Number
- +varchar: Color
- +varchar: year
- +varchar: Trim
- +int: Odometer
- +date: Last Service
- +date: Next Service
9. User Interface

Login Module

User

Driver Module

Manager Module

The user interface is the application, from the point of view of the users. Do your classes and their interactions (the logical and process views) impose restrictions on the user interface? Would removing some of these restrictions improve the user interface? Use some form of user interface flow model to provide an overview of the UI steps and flows. Don't go into too much refinement. You should include screenshots or wireframe layouts of significant pages or dialog elements. Make sure to indicate which of the system level modules or components that each of these user interface elements is interacting with.

9.1 Overview of User Interface
Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback Information that will be displayed for the user. This is an overview of the UI and its use. The user manual will contain extensive detail about the actual use of the software.
9.2 Screen Frameworks or Images
These can be mockups or actual screenshots of the various UI screens and pop ups.

9.3 User Interface Flow Model
A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.

9.4 Requirement

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Component</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.4.1 Positional data collected shall include the following, longitude, latitude and timestamp. Positional data in this context refers to global satellite positioning which will be used in conjunction with trip data. The timestamp shall have a date and the time from a 24 hour clock.</td>
<td></td>
<td>Hardware Controlled, by Ublox Software. See Section</td>
</tr>
</tbody>
</table>

9.4.2 Vehicle Diagnostics data will be made up of speed, current odometer, and all pertinent data which is collected by the data ingestion device. This data should be all the sensors that exist on the car. Discretion will be used to determine what sensor data does not need to be stored. This data.
will be used to determine if the vehicle has any operating abnormalities. Vehicle diagnostics data will be stored within a data lake for later data analysis. The analysed data will be available for maintenance and mechanics.

<table>
<thead>
<tr>
<th>9.4.3 LACFMS shall display the path traveled on a map which will be accessible by the management user. The path or route shall display the velocity of the vehicle along each route traveled.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>9.4.4 The system shall interconnect microcontrollers with appropriate technology to perform high bandwidth transactions for proper data ingestion, storage and transmission of stored data.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>9.4.4 The system shall require driver validation.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>9.4.5 The driver of a Los Angeles County vehicle shall sign in to a vehicle by swiping their identification badge. The authentication system may disable the vehicle from starting.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>9.4.6 All data will be sent to a Los Angeles County server infrastructure and stored into a database. Vehicle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnostic Data will be saved to a data lake. Trip Data will be saved to a relational database. Trip Data must be queryable by applications for supervisors and managers.</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>9.4.7 All employees shall input specific trip information into the web application.</td>
</tr>
<tr>
<td>9.4.8 The LACFMS system shall have a web application which shall register new employees if they do not currently exist in the database.</td>
</tr>
<tr>
<td>9.4.9 The LACFMS system shall display GPS data with vehicle locations.</td>
</tr>
<tr>
<td>9.4.10 All data gathered from driver will be stored in a relational database such as SQL.</td>
</tr>
<tr>
<td>9.4.11 Employees must be identified with each vehicle that they have driven.</td>
</tr>
<tr>
<td>9.4.12 Installation of LACFMS devices must not splice into a vehicle’s wiring.</td>
</tr>
<tr>
<td>9.4.13 The system should allow for the generation of vehicle usage history for any given time period.</td>
</tr>
</tbody>
</table>
Create a table that lists each of the requirements that were specified in the SRS document for this software.

For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.

For each entry describe the method for testing that the requirement has been met.
### 11. Data Dictionary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAN</td>
<td>A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer.</td>
</tr>
<tr>
<td>COTS (Commercial off the shelf)</td>
<td>Refers to ready-made merchandise that is available for sale, defined by market need, significant functionality and complexity, and self-contained.</td>
</tr>
<tr>
<td>ECU</td>
<td>The Parts of the Engine the ECU Controls. The ECU, also known as the car computer, provides controls for a variety of systems within the engine. The following sections will examine these systems, including the control of air:fuel ratio, ignition timing, and idle speed.</td>
</tr>
<tr>
<td>Global Positioning System</td>
<td>is a radio navigation system that allows land, sea, and airborne users to determine their exact location, velocity, and time 24 hours a day, in all weather conditions, anywhere in the world.</td>
</tr>
<tr>
<td>Graphical user interface (GUI /ˈɡuː.i/)</td>
<td>is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text-based user interfaces, typed command labels or text navigation.</td>
</tr>
<tr>
<td>Internet of Things (IoT)</td>
<td>The internet of things is the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive.</td>
</tr>
<tr>
<td><strong>LACFMS</strong></td>
<td>Los Angeles County Fleet Management System (LACFMS): Describes the overall software in this document.</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Low Frequency (LF) RFID</strong></td>
<td>The LF band covers frequencies from 30KHz to 300KHz. Typically LF RFID systems operate at 125 KHz, although there are some that operate at 134 KHz. This frequency band provides a short read range of 10 cm, and has slower read speed than the higher frequencies, but is not very sensitive to radio wave interference.</td>
</tr>
<tr>
<td><strong>MQTT Protocol Specifications</strong></td>
<td>$MQTT$ v3.1.1 is an OASIS Standard. ... $MQTT$-SN is a publish/subscribe messaging protocol for wireless sensor networks (WSN), with the aim of extending the $MQTT$ protocol beyond the reach of TCP/IP infrastructure for Sensor and Actuator solutions</td>
</tr>
<tr>
<td><strong>OBD-II (On-board diagnostics Parameter IDs)</strong></td>
<td>are codes used to request data from a vehicle, used as a diagnostic tool. SAE standard J/1979 defines many PIDs, but manufacturers also define many more PIDs specific to their vehicles.</td>
</tr>
<tr>
<td><strong>TBD</strong></td>
<td>To be determined</td>
</tr>
</tbody>
</table>
12. Hardware Schematics

Neo-6M RPI
VCC to Pin 1, which is 3.3v
TX to Pin 10, which is RX (GPIO15)
RX to Pin 8, Which is TX (GPIO14)
Gnd to Pin 6, which is Gnd

1. ELM327 connect to OBD II interface
2. ELM327 connected to Raspberry Pi using usb.

Communication
Bluetooth to Raspberry Pi
USB to Raspberry Pi
13. References

12.0 IEEE specification 1016-2009

12.1 OBD II 327 ELM Datasheet

12.2 Neo 6M GPS Hardware Datasheet

12.3 Raspberry PI3 Model B Datasheet

12.4 Apache Nifi Documentation
https://nifi.apache.org/docs.html